CSCE 350

Project 3

Alan Achtenberg

A simple MIPS Processor Extended

Truth Tables

Main Control

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPCODE | RegDst | ALUSrc | MemToReg | RegWr | MemR | MemW | Branch | Jump | SignExt | ALUOp |
| RTYPE 000000 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | X | 1111 |
| ADDI 001000 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0111 |
| ADDIU 001001 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0001 |
| NOP 000000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1111 |
| LW 100011 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0111 |
| SW 101011 | X | 1 | X | 0 | 0 | 1 | 0 | 0 | 1 | 0111 |
| BEQ 000100 | X | 0 | X | 0 | 0 | 0 | 1 | 0 | X | 0010 |
| BNE 000101 | X | 0 | X | 0 | 0 | 0 | 1 | 0 | X | 1001 |
| JUMP 000010 | X | 0 | X | 0 | 0 | 0 | 0 | 1 | X | X |
| ORI 001101 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0101 |
| ANDI 001100 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0100 |
| SLTI 001010 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1010 |
| SLTIU 001011 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1011 |
| XORI 001110 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0110 |

ALU Control

|  |  |  |
| --- | --- | --- |
| ALUOp | FUNCT | ALU Input |
| FUNCT 1111 | INST[3:0]=XXXX | XXXX |
| ADD | Don’t care | 0111 |
| ADDU | Don’t care | 0001 |
| SUB | X | 0010 |
| SUBU | X | 0011 |
| AND | X | 0100 |
| OR | X | 0101 |
| XOR | X | 0110 |
| SLT | X | 1010 |
| SLTU | X | 1011 |
| NOP | X | 0000 |
| SLL | X | 1000 |
| SRL | X | 1100 |
| SRA | X | 1110 |
| BNE | X | 1001 |

Modules

New module[ for Project 3 Data Memory

Similar to the register file except it is much larger and only writes to and reads from registers.

ALU Control

Extended for project three to include support for memory instructions as well as branch and jump instructions.

]

Program Counter

Simply increases The Current PC by four after every clock cycle. If the Reset\_L has a rising edge the program counter is reset to start PC.

Register File

Contains the registers for processor operations. It always selects 2 registers to read and when enabled it writes back results from the ALU to the Register File. When Reset\_L has a rising edge the Register File clears the registers to 0.

Processor Control and ALU Control

Takes in the Intruction and from the OP code it determines the control lines for the muxes implemented in SingleCycleProc. When a OP code of 000000 is given the control unit passed on a 1111 (FUNCT code) to the ALU Control which allows the ALU to determine what the correct ALU OP is, For other instruction types the control

SingleCycleProc

Combines all the modules using several MUX that are controlled by the Processor Control.

The Critical Path is when writing to the memory because of slow access time of the data mem registers.

To run the program simply input vcs –R SingleCycleProc.v into the console, all necessary files are included. Currently register writes are displayed as well as branches taken and words stored back to memory.

Final output of SingleCycleProc.v with imeminit\_simple\_test.v is below

----------------------------------

REGWRITE Reg 5 current 0 new 0

----------------------------------

----------------------------------

REGWRITE Reg 2 current 0 new 0

----------------------------------

----------------------------------

REGWRITE Reg 3 current 3 new 3

----------------------------------

----------------------------------

REGWRITE Reg 3 current 6 new 6

----------------------------------

----------------------------------

REGWRITE Reg 3 current 12 new 12

----------------------------------

----------------------------------

REGWRITE Reg 7 current 4 new 4

----------------------------------

----------------------------------

REGWRITE Reg 6 current 50 new 50

----------------------------------

----------------------------------

REGWRITE Reg 5 current 50 new 50

----------------------------------

----------------------------------

REGWRITE Reg 2 current 4 new 4

----------------------------------

BRANCH Next 24

----------------------------------

REGWRITE Reg 6 current 40 new 40

----------------------------------

----------------------------------

REGWRITE Reg 5 current 90 new 90

----------------------------------

----------------------------------

REGWRITE Reg 2 current 8 new 8

----------------------------------

BRANCH Next 24

----------------------------------

REGWRITE Reg 6 current 30 new 30

----------------------------------

----------------------------------

REGWRITE Reg 5 current 120 new 120

----------------------------------

----------------------------------

REGWRITE Reg 2 current 12 new 12

----------------------------------

BRANCH Next 24

BRANCH Next 44

STORE WORD old MEM[ 5]= 0 new data= 120

----------------------------------

REGWRITE Reg 0 current 0 new 0

----------------------------------

----------------------------------

REGWRITE Reg 0 current 0 new 0

----------------------------------

----------------------------------

REGWRITE Reg 1 current 1 new 1

----------------------------------

----------------------------------

REGWRITE Reg 2 current 4294967295 new 4294967295

----------------------------------

SLT:- [ 2800] tmp = 4294967295 [11111111111111111111111111111111]; Cout=0, Ovf=0; A=4294967295, B= 0

SLT:+R= 1 [00000000000000000000000000000001]

----------------------------------

REGWRITE Reg 5 current 1 new 1

----------------------------------

----------------------------------

REGWRITE Reg 6 current 2 new 2

----------------------------------

----------------------------------

REGWRITE Reg 7 current 3 new 3

----------------------------------

----------------------------------

REGWRITE Reg 8 current 4294967294 new 4294967294

----------------------------------

----------------------------------

REGWRITE Reg 9 current 2 new 2

----------------------------------

STORE WORD old MEM[ 8]= 0 new data= 2

----------------------------------

REGWRITE Reg 0 current 0 new 0

----------------------------------

----------------------------------

REGWRITE Reg 10 current 1 new 1

----------------------------------

----------------------------------

REGWRITE Reg 11 current 2 new 2

----------------------------------

----------------------------------

REGWRITE Reg 12 current 4294967295 new 4294967295

----------------------------------

----------------------------------

REGWRITE Reg 13 current 2 new 2

----------------------------------

----------------------------------

REGWRITE Reg 14 current 0 new 0

----------------------------------

----------------------------------

REGWRITE Reg 15 current 0 new 0

----------------------------------